

CSC 34200 - Computer Organization

Instructor: Prof. Zheng Peng

Lab 04

- The entire source code for the project is shown below:

```

1      .data                                #data segment
2      nl: .ascii "\n"                     #ASCII for a new line
3      .align 2                             #aligned at word boundary
4      course: .ascii "CSC343: "           #course name to be displayed
5      .align 2                             #aligned at word boundary
6      name: .ascii "Samuel Youssef "      #name
7      .align 2                             #aligned at word boundary
8      AndWord: .ascii "and "              #"and" word to be displayed
9      .align 2                             #aligned at word boundary
10     ID: .ascii "ID : 23402306"           #ID number to be displayed
11     .align 2                             # aligned at word boundary
12     input: .ascii "Please enter a Fibonacci index number " #statement at input to guide the user through input process
13     .align 2                             #aligned at word boundary
14     exitSmt: .ascii "(0 will stop the program) : " #statement to appear at input to guide the user to exit the program
15     .align 2                             #aligned at word boundary
16     leftPar: .ascii "("                 #left parenthesis
17     .align 2                             #aligned at word boundary
18     rightPar: .ascii ")"               #right parenthesis
19     .align 2                             #aligned at word boundary
20     Fsymbol: .ascii "F"                 #symbol of F to symbolize that it is a fibonacci number
21     .align 2                             #aligned at word boundary
22     bye: .ascii "bye"                   #bye to appear when the program is about to exit
23     .align 2                             #aligned at word boundary
24     toolarge: .ascii "the number is too large" #too large number message to be printed out
25     .align 2                             #aligned at word boundary
26     negative: .ascii "the index is a negative number please try again" #index entered is a negative number
27     .align 2
28
29     .text                                #code segment
30     .globl main                          #declare main to be global
31
32     main:
33         li $v0, 4                        #print string
34         la $a0, course                   #load course name
35         syscall                          #calling syscall to print string
36
37         li $v0, 4                        #print string
38         la $a0, name                     #loading name string to be printed
39         syscall                          #calling syscall to print string
40
41         li $v0, 4                        #print string
42         la $a0, AndWord                  #loading AndWord string to be printed
43         syscall                          # calling syscall to print string
44
45         li $v0, 4                        #print string
46         la $a0, ID                       #loading the ID string to be printed
47         syscall                          #calling syscall to print string
48
49         li $v0, 4                        #print string
50         la $a0, nl                       #loading new line string to be printed
51         syscall                          #calling syscall to print string
52
53         addiu $t6, $zero, 1              #setting $t6 to 1 for comparison
54
55     loopInput: li $v0, 4                  #print string
56                 la $a0, input             #loading input statement to be printed on the screen
57                 syscall                   #calling syscall to print string
58
59                 li $v0, 4                  #print string
60                 la $a0, exitSmt           #loading directions on how to exit the program to be printed on the screen
61                 syscall                   #calling syscall to print string
62
63                 li $v0, 5                  #reading the index of fibonacci number
64                 syscall                   #calling syscall to input index number
65
66                 move $t1, $v0              #saving the index number of fib number
67                 move $a0, $v0              #preparing for function call by setting argument a0 for function fibonacci
68

```

```

69      beq $v0, $zero, exit1          #if input is equal to zero, the program exits
70      slt $t7, $v0, $zero           #if index number inputted is negative
71      bne $t7, $t6, notNegative     #the index is not negative, proceed
72
73      li $v0, 4                      #system call to print string
74      la $a0, Fsymbol               #loading the address of leftPar to be printed
75      syscall                       #call to OS
76
77      li $v0, 4                      #system call to print string
78      la $a0, leftPar               #loading the address of leftPar to be printed
79      syscall                       #call to OS
80
81      li $v0, 1                      #system call to print integer
82      la $a0, ($t1)                 #loading address of index of fib number
83      syscall                       #call to OS
84
85      li $v0, 4                      #system call to print string
86      la $a0, rightPar              #loading the address of rightPar
87      syscall                       #call to OS
88
89      li $v0, 4                      #system call to print string
90      la $a0, negative               #loading the address of rightPar
91      syscall                       #call to OS
92
93      li $v0, 4                      #print string
94      la $a0, nl                     #loading new line string to be printed
95      syscall                       #calling syscall to print string
96      j continueProc               #continue input loop process
97
98
99
100
101 notNegative:
102     jal fibonacci2                 #calling the function
103
104     move $t0, $v0                  #move return value of v0 to t0
105     slt $t2, $v0, $zero            #if the return number is 0xffffffff which is -1 is less than zero
106     beq $t2, $zero, repeatloop     #jump directly to repeat loop to repeat the loop with printing out fib number.
107
108     li $v0, 4                      #system call to print string
109     la $a0, Fsymbol               #loading the address of leftPar to be printed
110     syscall                       #call to OS
111
112     li $v0, 4                      #system call to print string
113     la $a0, leftPar               #loading the address of leftPar to be printed
114     syscall                       #call to OS
115
116     li $v0, 1                      #system call to print integer
117     la $a0, ($t1)                 #loading address of index of fib number
118     syscall                       #call to OS
119
120     li $v0, 4                      #system call to print string
121     la $a0, rightPar              #loading the address of rightPar
122     syscall                       #call to OS
123
124     li $v0, 4                      #system call to print string
125     la $a0, toolarge              #loading the address of rightPar
126     syscall                       #call to OS
127
128     li $v0, 4                      #print string
129     la $a0, nl                     #loading new line string to be printed
130     syscall                       #calling syscall to print string
131     j continueProc               #continue input loop process
132
133 repeatloop:
134     li $v0, 4                      #system call to print string
135     la $a0, Fsymbol               #loading the address of leftPar to be printed
136     syscall                       #call to OS

```

```

137
138      li $v0, 4                #system call to print string
139      la $a0, leftPar          #loading the address of leftPar to be printed
140      syscall                  #call to OS
141
142      li $v0, 1                #system call to print integer
143      la $a0, ($t1)            #loading address of index of fib number
144      syscall                  #call to OS
145
146      li $v0, 4                #system call to print string
147      la $a0, rightPar         #loading the address of rightPar
148      syscall                  #call to OS
149
150
151      li $v0, 1                #print integer
152      la $a0, ($t0)            #moving address of fib number to be printed out
153      syscall                  #making the system call
154
155      li $v0, 4                #print string
156      la $a0, nl               #loading new line string to be printed
157      syscall                  #calling syscall to print string
158
159  continueProc:
160      j loopInput              #call to repeat the loop for the user to input another index of fib number
161
162
163  exit1:
164      li $v0, 4                #print string
165      la $a0, bye              #loading bye message to be printed on the screen
166      syscall                  #calling syscall to print string
167      li $v0, 10               #system call type 10, standard exit
168      syscall                  #call to OS
169
170
171  #registers used inside this function
172  #a0 to store the argument of the function
173  #s0 to store constant 1
174  #s1 to store constant zero
175  #s2 to store constant 100
176  #s3 to store constant -1
177  #s4 to store boolean value of comparison
178  #s5 to store the result of addition between f[n-1] and f[n-2]
179  #s6 to store the value of v0
180  #s7 to store the value of v1
181  #-----Fibonacci2 function-----
182
183  .globl fibonacci2            #making the function print global to be accessed by other files
184  fibonacci2:                  #label function
185
186      addi $sp, $sp, -40        #adding space for register used in the function on the stack
187      sw $ra, 36($sp)           #saving register ra on the stack
188      sw $a0, 32($sp)           #saving a0 on the stack
189      sw $s7, 28($sp)           #saving register s7 on the stack
190      sw $s6, 24($sp)           #saving register s6 on the stack
191      sw $s5, 20($sp)           #saving register s5 on the stack
192      sw $s4, 16($sp)           #saving register s4 on the stack
193      sw $s3, 12($sp)           #saving register s3 on the stack
194      sw $s2, 8($sp)            #saving register s2 on the stack
195      sw $s1, 4($sp)            #saving register s1 on the stack
196      sw $s0, 0($sp)           #saving register s0 on the stack
197
198      addiu $s0, $zero, 1       #setting s0 equal to 1 for comparison
199      move $s1, $zero           #setting s1 to zero, later used in comparison
200      addiu $s2, $zero, 100     #setting s2 to 100 for comparison
201      addiu $s3, $zero, -1      #setting s3 = -1 to be used later
202      move $s4, $zero           #moving zero to register s4 to be later used in comparison
203      move $s5, $zero           #s5 used in the addition process inside the recursive function
204

```

```

205
206
207     bne $a0, $s1, label1          #branch to label1 if a0 != 0
208     move $v0, $zero               #move to v0 (later assigned to v0) zero
209     move $v1, $zero               #move to v1 (later assigned to v1) zero
210     lw $a0, 32($sp)                #restoring register a0 from the stack
211     lw $ra, 36($sp)                #restoring register ra from the stack
212     addi $sp, $sp, 40              #restoring the stack pointer
213     jr $ra                         #return
214
215 label1:
216     bne $a0, $s0, label2          #branch to label2 if a0 != 1
217     move $v0, $s0                 #setting v0 to 1
218     move $v1, $zero               #setting v1 to 0
219     lw $a0, 32($sp)                #restoring register a0 from the stack
220     lw $ra, 36($sp)                #restoring register ra from the stack
221     addi $sp, $sp, 40              #restoring the stack pointer
222     jr $ra                         #return
223
224 label2:
225     sgtu $s4, $a0, $s2             #s4 == 1 if a0 == 100
226     bne $s4, $s0, label3          #branch to label3 if a1 != 100
227     move $v0, $s3                 #setting v0 to value of s3 which is -1
228     lw $a0, 32($sp)                #restoring register a0 from the stack
229     lw $ra, 36($sp)                #restoring register ra from the stack
230     addi $sp, $sp, 40              #restoring the stack pointer
231     jr $ra                         #return
232
233 label3:
234     addiu $a0, $a0, -1             #decrementing a0
235     jal fibonacci2                 #recursive call
236
237     move $s6, $v0                  #getting the F[n-1]
238     move $s7, $v1                  #getting the F[n-2]
239     beq $s6, $s3, exit2            #exiting if the index of fibonacci number is out of bounds that we can calculate (>100)
240     addu $s5, $s6, $s7             #adding the two elements F[n-1] and F[n-2] to get F[n]. value of F[n] is saved in s5
241     slt $s4, $s5, $zero            #checking if there is overflow as a result of the addition.
242     bne $s4, $zero, label4         #if there is overflow jump to label4
243     move $s7, $s6                  #setting the correct value for v0
244     move $s6, $s5                  #setting the correct value for v1
245
246     j exit2                        #jumping to exit2 after calculating new F[n] and F[n-1] for return
247
248 label4:
249
250     move $v0, $s3                  #setting v0 to value of s3 which is -1
251     lw $a0, 32($sp)                #restoring register a0 from the stack
252     lw $ra, 36($sp)                #restoring register ra from the stack
253     addi $sp, $sp, 40              #restoring the stack pointer
254     jr $ra                         #return
255
256 exit2:
257
258     move $v0, $s6                  #putting the final value of $v0 in $v0
259     move $v1, $s7                  #putting the final value of $v1 in $v1
260     lw $s0, 0($sp)                 #restoring register s0 from the stack
261     lw $s1, 4($sp)                 #restoring register s1 from the stack
262     lw $s2, 8($sp)                 #restoring register s2 from the stack
263     lw $s3, 12($sp)                #restoring register s3 from the stack
264     lw $s4, 16($sp)                #restoring register s4 from the stack
265     lw $s5, 20($sp)                #restoring register s5 from the stack
266     lw $s6, 24($sp)                #restoring register s6 from the stack
267     lw $s7, 28($sp)                #restoring register s7 from the stack
268     lw $a0, 32($sp)                #restoring register a0 from the stack
269     lw $ra, 36($sp)                #restoring register ra from the stack
270     addi $sp, $sp, 40              #restoring the stack pointer
271     jr $ra                         #returning to caller
272

```

Figure 1: Entire source code for the project.

- A run for the program is shown below:

```
CSC343: Samuel Youssef and ID : 23402306
Please enter a Fibonacci index number (0 will stop the program) :1
F(1) =1
Please enter a Fibonacci index number (0 will stop the program) :2
F(2) =1
Please enter a Fibonacci index number (0 will stop the program) :4
F(4) =3
Please enter a Fibonacci index number (0 will stop the program) :12
F(12) =144
Please enter a Fibonacci index number (0 will stop the program) :30
F(30) =832040

Please enter a Fibonacci index number (0 will stop the program) :45
F(45) =1134903170
Please enter a Fibonacci index number (0 will stop the program) :46
F(46) =1836311903
Please enter a Fibonacci index number (0 will stop the program) :47
F(47) =the number is too large
Please enter a Fibonacci index number (0 will stop the program) :100
F(100) =the number is too large
Please enter a Fibonacci index number (0 will stop the program) :1000
F(1000) =the number is too large
Please enter a Fibonacci index number (0 will stop the program) :-1
F(-1) =the index is a negative number please try again
Please enter a Fibonacci index number (0 will stop the program) :-1000
F(-1000) =the index is a negative number please try again
Please enter a Fibonacci index number (0 will stop the program) :0
bye
-- program is finished running --
```

Figure 2: A run of the program.